Car-Sharing service

PowerEnJoy

Domain Properties

- The credentials and payment information provided by the users are correct.
- Assume that the credits in the users credit cards or bank accounts are always enough.
- Each car and safe area and users' mobile devices contain GPS and the GPS works correctly.
- Each car has a way to determine the number of people into the car.
- Each car knows when to lock the car after user leaves it.
- Each car knows its own battery's state(percent of battery empty,charging).
- Users can not leave the car while driving.
- Cars are always connected to the network.
- The system always sends the message to user when it is necessary.
- There is a timer to take care of the management of some operations.
- There are maintainers to take care of charging the battery of cars.
- There are maintainers in the stations.
- There will be no car accident which interrupts the users while driving

Glossary

Some glossaries are interpreted here

- Guest
  - Guests are people who haven't performed the log in operation.They may be registered clients or unregistered clients. As long as they have not performed log in, they are guests.
- User
  - Refer to the clients who have performed log in,
- Credentials
  - Some personal information about each user, including:
  - Name
  - E_mail address
  - Phone number
  - Number of drive license

.....

Functional Requirements

According to the goals, we can derive 13 requirements

……

Goal [1]

[G1]Users can register to the system and get the password by providing credentials and payment information

- The system must let only one registration per credential.
- The system must be able to check whether the payment information is correct or not.
- The system must create and maintain only one password per credential
- The system must check whether the password is correct under the credential
- The system must allow users to login when provided with the correct credential and password

Goal[2]

[G2]Registered users can find the locations of available cars within a certain distance from their current location or from a specified address

- The system must be able to get all the locations of the cars
- The system has a default range which identifies the term "nearby"
- The system must be able to get all the information of the nearby cars with a certain locaiton as input

Goal[3]

[G3]Among the available cars in a certain geographical region, users must be able to reserve a single car for up to one hour before they pick it up.

- The system must allow users to reserve a single car
- The system maintains a clock for each car. Once the car is resvered by a user, a one-hours time clock starts

Goal[4]

[G4]If a car is not picked-up within one hour from the reservation, the system tags the car as available again, and the reservation expires; the user pays a fee of 1 EUR.

- The system must allow the user who has made a reservation for a car to stop the clock of that car within one hour picking up time
- The system must charge the user 1 euro for not being able to stop the clock within one hour
- The system will make the car state changed to available if the clock is not stopped within in one hour

Goal[5]

[G5]A user that reaches a reserved car must be able to tell the system she’s nearby, so the system unlocks the car and the user may enter.

- The system must be able to check the position of the user according to the user's GPS
- The system must be able to check the position of the car according to the car's GPS
- The system should be able to receive the messages which are sent by different users as the same time
- The system should be able to check whether the postition of the user is the same as the corresponding car
- If the user and the corresponding car are in the same position, the system must transfer the control information to the appropriate car for unlocking the car
- If the user and the corresponding car are not in the same position, the system must be able to keep the car locked

Goal[6]

[G6]As soon as the engine ignites, the system starts charging the user for a given amount of money per minute; the user is notified of the current charges through a screen on the car.

- The system must be able to detect whether the engine of the car is ignited or not
- The system must be able to record the total using time
- The system must be able to estimate the total fee during the renting service
- The system must be able to transfer the charging information to the appropriate car

Goal[7]

[G7]The system stops charging the user as soon as the car is parked in a safe area and the user exits the car; at this point, the system locks the car automatically.

- The system must be able to check the position of the car according to the car's GPS
- The system must be able to detect whether the car is parked or not
- The system must be able to check whether the car is parked in the safe area
- The system must be able to detect whether the user has exited the car
- if the user still stay in the car, the system must be able to keep the car in the unlocked state
- if the car is not parked in the safe area, the system must be able to keep the car in the unlocked state and keep charging
- The system must let the user know whether the car is in the safe area
- if the car is parked in the safe area and the user has exited the car, the system must be able to transfer the control information for locking the car

Goal[8]

[G8]The set of safe areas for parking cars is pre-defined by the management system.

- The system must be able to insert the area in the safe area set
- The system must be able to delete the area in the safe area set
- The system must be able to retrieve the area in the safe area set

Goal[9]

[G9] If the system detects the user took at least two other passengers onto the car, the system applies a discount of 10% on the last ride.

- The system will analyze whether more than two passengers, excluding driver, onto the cars or not, if it actually goes beyond, the system will register a discount of 10% on the last ride

Goal[10]

[G10]If a car is left with no more than 50% of the battery empty, the system applies a discount of 20% on the last ride.

- After the lock of car, the system will examine the level of battery, if over half the battery is remained, then the system will register a discount of 20% on the last ride

Goal[11]

[G11]If a car is left at special parking areas where they can be recharged and the user takes care of plugging the car into the power grid, the system applies a discount of 30% on the last ride.

- system will register a discount of 30% on the last ride if it recognize the car is parked at special area, and it's charging the battery within a determined time.

Goal[12]

[G12]If a car is left at more than 3 KM from the nearest power grid station or with more than 80% of the battery empty, the system charges 30% more on the last ride to compensate for the cost required to re-charge the car on-site.

- system must be able to get the positions of the cars and all positions of power grid stations.
- system will record an increase of 30% on the fee of last ride, if there are not any power grid stations within 3KM from the position of car or the car is left with no more than 20% of battery.
- system must inform the maintainers to retrive the cars with low battery to the stations.

Goal[13]

[G13]If the user enables the money saving option, he/she can input his/her final destination and the system provides information about the station where to leave the car to get a discount. This station is determined to ensure a uniform distribution of cars in the city and depends both on the destination of the user and on the availability of power plugs at the selected station.

- The system must be able to get the distribution of the cars
- The system must be able to detect whether the user enables the money saving option or not
- The system must be able to select the station according to the user's destination for ensuring a uniform distribution of the cars
- The system must be able to detect the availability of power plugs at the selected station

Non-functional Requirements

From renting system, we set some non-functional requirements, including the user interface and so on ……

User Interface (1)

User Interface(2)

Scenario Identifying

For this system, we design some scenarios ……

Scenario 1

Mario is a college student who possesses a driving licence. He saw the ads about PowerEnJoy and decided to try this service. He uses his laptop to access the website of PowerEnjoy and proceeds the registering procedure. He provides his credentials and payment information. After the system analyzes his data, Mario receives a password. He logs into the system and finds some available cars near his place. He selects one car and makes a reservation. 30 minutes later, Mario arrives at the car position and he uses his laptop to notify the system that he is near the car. The system checks his position and unlocks the car. Mario drives home and on the car screen he can see his current driving route,charges and safe areas nearby. He stops the car in one of the safe areas near home and finishes his journey.

Scenario 2

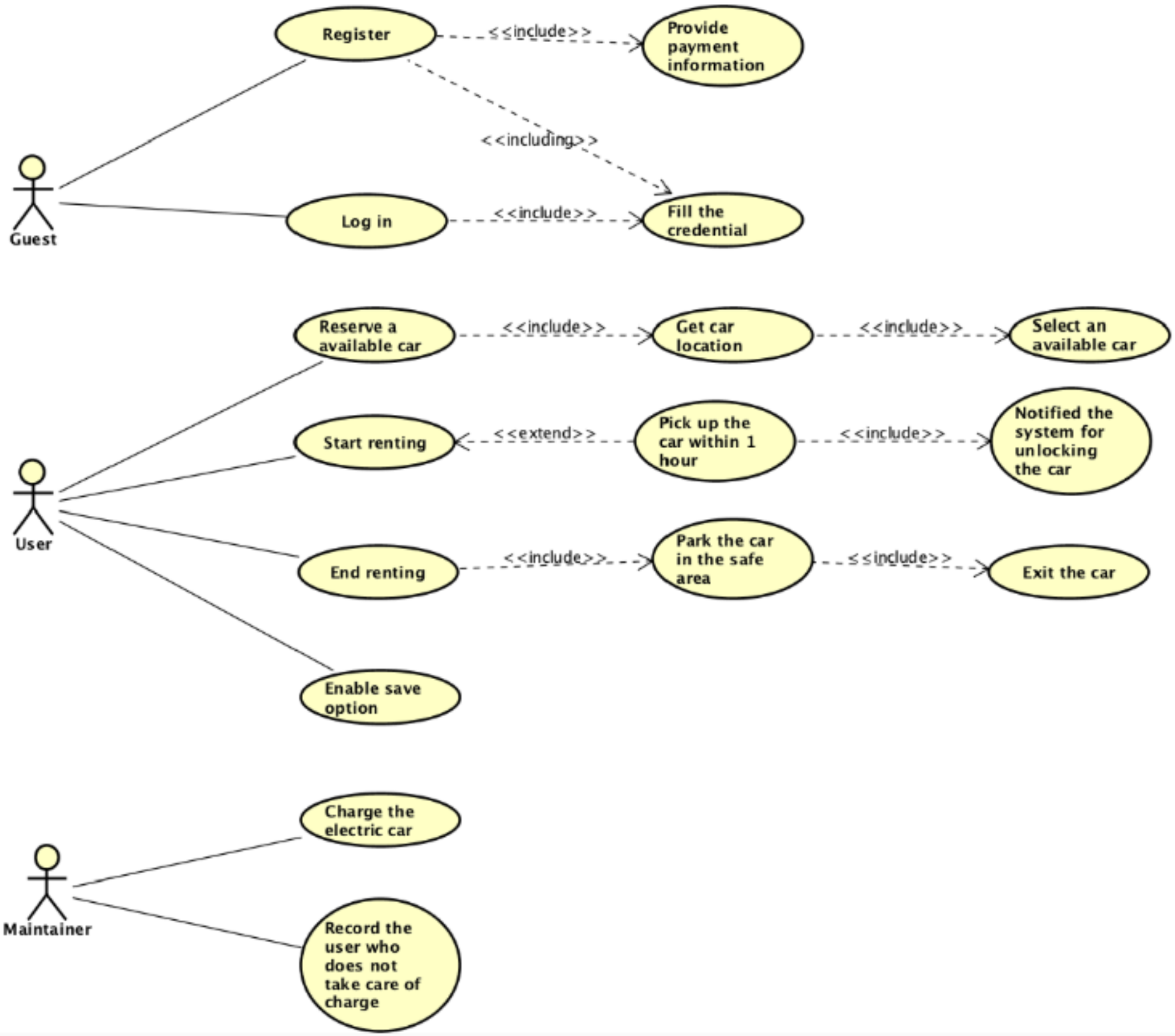
Nino is a college student. Meanwhile, he is also a user of the PowerEnjoy. He students in the university together with two of his roommates. Finishing his lectures at the afternoon, Nino comes back home together with his roommates. First of all, Nino login to the PowerEnJoy system using his smart phone. Then, he reserve a car and arrive at the position of the car. Nino pick it up and drive the car home. 40 minutes later, Nino stops the car at one the safe locations near his home. Since there are 2 passengers in the car in this renting service, Nino gets 10% discount for the total renting fee.

Scenario 3

Matteo is a college student who lives in a university residence, where is placed into another city different from his parents' one. At Christmas holiday, he decide to go back to parents' home with PowerEnjoy. After being rested and logged in the system, he reserves a car extremely close to the residence. He take immediately the car and drive at home. He arrives almost at 22:00, unfortunately, there is not any safe area with power plugs within 3km from his home, and the battery is less than 15%. So Matteo parks the car at near safe area without plugs. Due to a scarce battery remained and large distance from power plugs, Matteo pay 30% more on the trip.

Use Case

Now, let’s turn to some programmable parts ……


Register
<<include>>
Provide payment information
<<including>>
Guest
Log in
<<include>>
Fill the credential
Reserve a available car
<<include>>
Get car location
<<include>>
Select an available car
Start renting
<<extend>>
Pick up the car within 1 hour
<<include>>
Notified the system for unlocking the car
User
End renting
<<include>>
Park the car in the safe area
<<include>>
Exit the car
Enable save option
Charge the electric car
Maintainer
Record the user who does not take care of charge

Use case description

let’s discuss descriptions of this use case (partial) ……

Guest registers in the system

Name: Guest registers in the system
Actor: Guest
Entry condition: No
Flow of events:

- The user enters the web home page or opens the mobile app and clicks "Sign up"
- The user fills in the credentials
- The user fills in the payment information
- The system gives the user a password for loging in

Exit condition: The user receives password from his/her email box.
Exceptions:

- The user misses information during filling blanks.
- The user has already registered with the same credentials.
- the user's credentials or payment information are not correct.

Exceptions handling: The user can either continue filling all the correct information or terminates the registering process.

User reserves an available car

Name: User reserves an available car
Actor: User
Entry condition: The user successfully login with his/her e_mail address and password
Flow of events:

- The user clicks on the "From current address" button or sets the "From specified address" box to the desired value.
- The user sets the "Certain distance" to the desired value.
- The system notifies the user with the available cars within the desired distance of desired address.
- The user selects one of the available cars
- The user clicks on the "Reserve this car" button.
- The system notifies the user for the reservation has been done.

Exit condition: The user receives the notification about the reservation is completed and clicks the buttom "Confirm".
Exception:

- There is no available car within desired distance of desired address.
- The user does not want to reserve any of the available car
- The desired car has been reserved before the user completes the reservation

Exception handing: The user can reset the desired distance and desired address, or select another available car. The user can also cancel this reservation.

User ends renting

Name:User ends renting
Actor:User
Entry condition:The user has started the renting successfully
Flow of events:

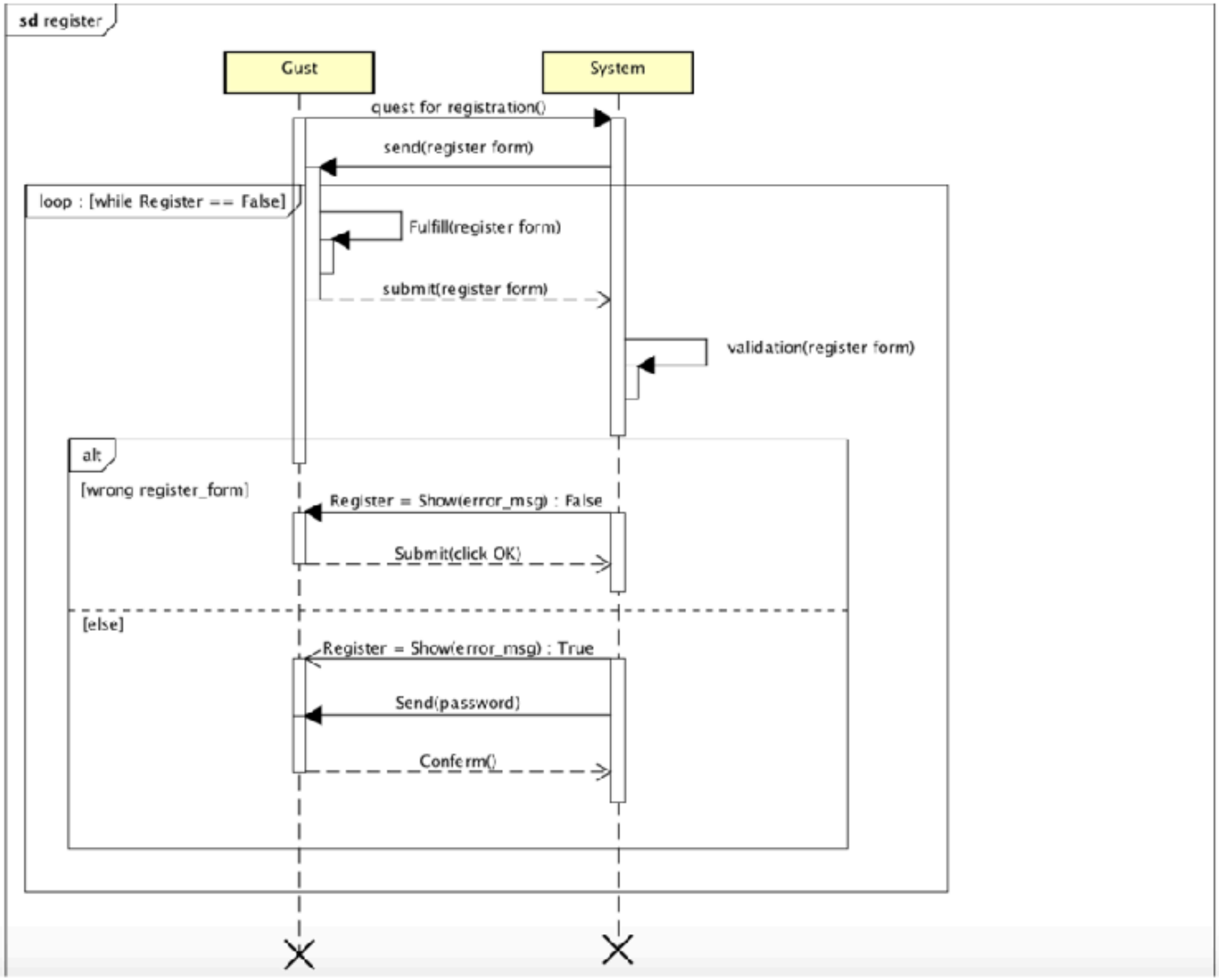
- The user parks the renting car in the safe area.
- The user exits the car.
- The system locks the car automatically.
- The system detects the passenger in this renting service, if the passenger is no less then 2, applies 10% discount for this renting service.
- The system detects the battery remaining amount, if the battery is no more than 50% empty, applies 20% discount for this renting service.
- The system detects whether the car is plugged into the power gird or not, if yes, applies 30% discount for this renting service.
- The system detects the position of the car, if the car is 3KM away from the power station and the battery is more than 80% empty, charges 30% more for this renting service.

Exit condition:The system calculate out the total fee for this renting servcie
Exception:The user does not park the car in the safe area.
Exception handing:The system notifies the user that he/she does not park the car in the correct position.

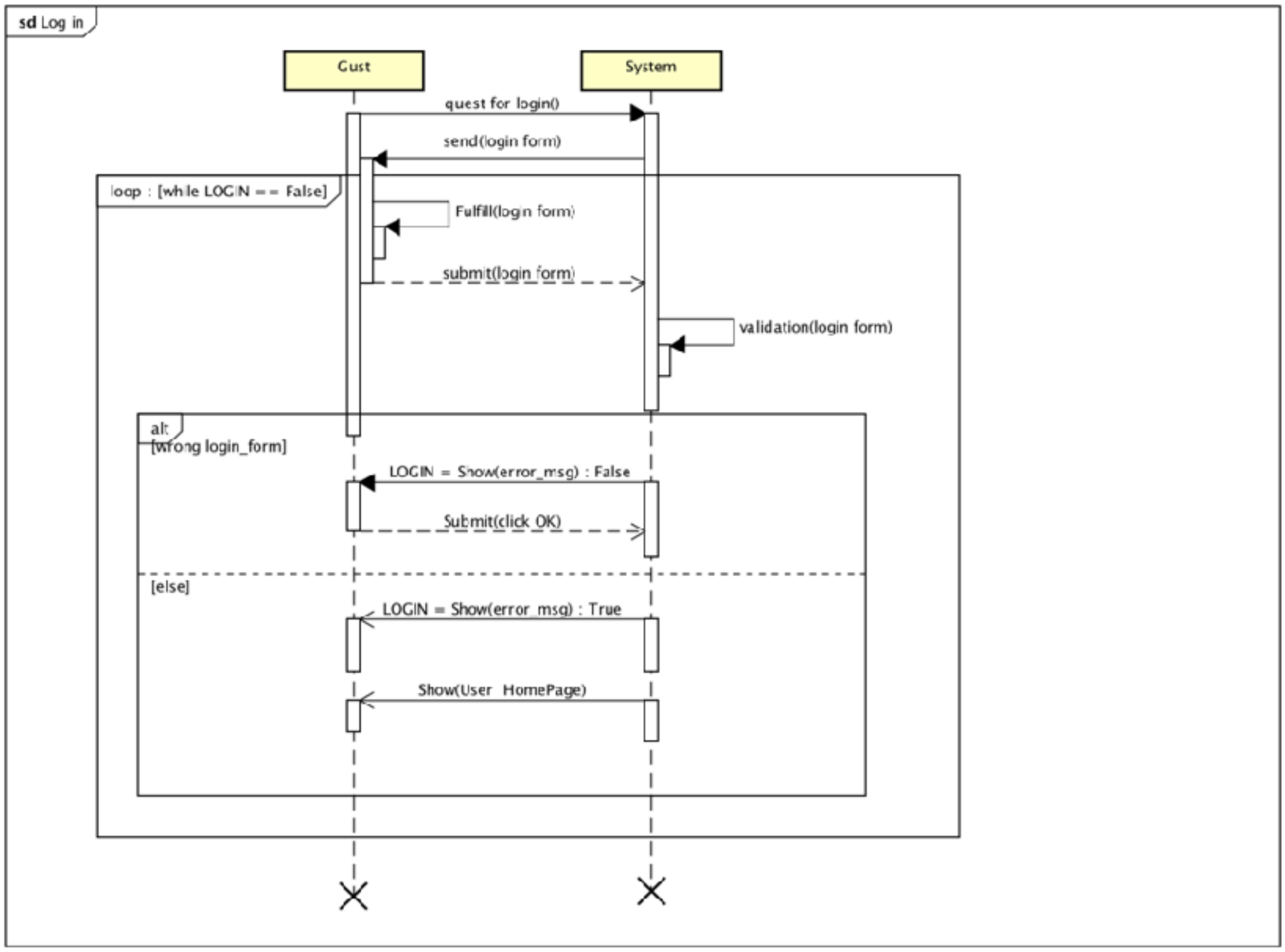
Sequence Diagram

After the definition of the use case, now we should focus on the sequence of the activities ……

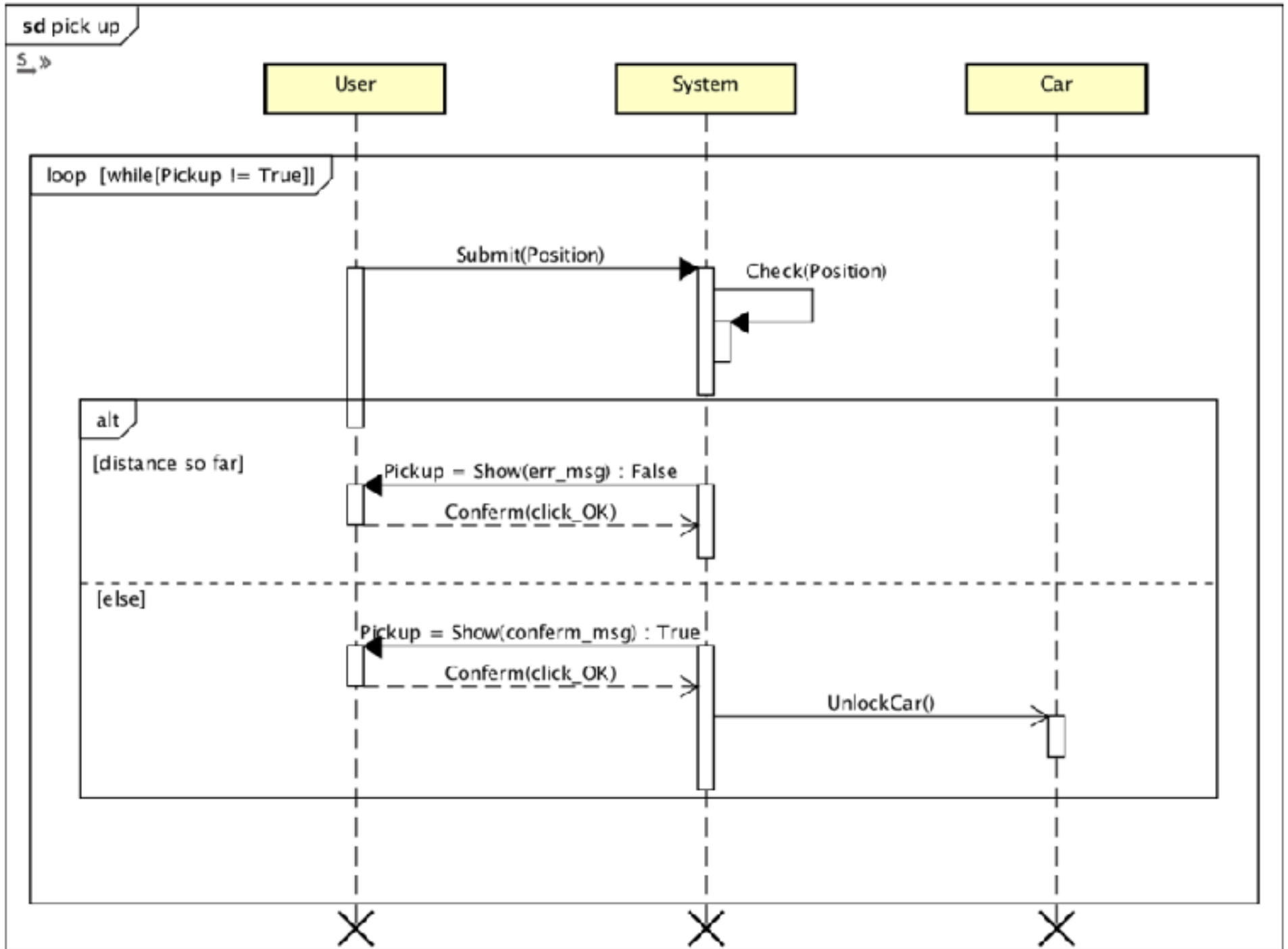
1. Register


sd register
Gust
System
quest for registration()
send(register form)
loop : [while Register == False]
Fulfill(register form)
submit(register form)
validation(register form)
alt
[wrong register_form]
Register = Show(error_msg) : False
Submit(click OK)
[else]
Register = Show(error_msg) : True
Send(password)
Conferm()

2. Log in


sd Log in
Gust
System
quest for login()
send(login form)
loop : [while LOGIN == False]
Fulfill(login form)
submit(login form)
validation(login form)
alt
[wrong login_form]
LOGIN = Show(error_msg) : False
Submit(click OK)
[else]
LOGIN = Show(error_msg) : True
Show(User HomePage)

3. Pick up the car



Alloy

This is the final part of the assignment 1

Part 1

```
sig Float { }
sig Astring { }

sig Credential {
    email : one Astring,
    password : one Astring,
    username : one Astring,
    drivingLicenseNo : one Astring,
}
sig PaymentInfo {
    name : one Astring,
    surname : one Astring,
    cardType : one Astring,
    cardNo : one Astring,
    cellphoneNo : one Astring,
    taxCode : one Astring
} {
    all p : PaymentInfo | p in User.paymentInfo
}

sig User {
    credential : one Credential,
    paymentInfo : set PaymentInfo,

} {
    paymentInfo != none
}

sig Position {
    latitude : one Float,
    longitude : one Float
} {
    all p : Position | p in Area.points
}
```

Part 2

```
abstract sig Area {
    points : set Position
}
sig SafeArea extends Area { }
sig UnSafeArea extends Area { }
sig Station extends SafeArea { }
sig NonStation extends SafeArea { }

abstract sig CarState { }
one sig AVAILABLE extends CarState { }
one sig RESERVED extends CarState { }
one sig WORKING extends CarState { }
one sig CHARGING extends CarState { }
one sig UNCHARGING extends CarState { }

abstract sig BatteryState { }
one sig FULL extends BatteryState { }
one sig MEDIUM extends BatteryState { }
one sig LOW extends BatteryState { }
```

```
sig Car {
    plate : one Astring,
    capacity : one Int,
    state : one CarState,
    battery : one BatteryState,
    currentPosition : one Position,
} {
    capacity > 0 and capacity <= 4
    state = AVAILABLE and state = RESERVED <=> battery = FULL
    state = AVAILABLE implies currentPosition in Station.points
    state = RESERVED implies currentPosition in Station.points
    state = CHARGING implies currentPosition in Station.points
    state = UNCHARGING implies currentPosition in NonStation.points
}

sig Reservation {
    user : one User,
    car : one Car,
} {
    car.state = RESERVED
}
```

Part 3

```
one sig System {
    users : set User,
    cars : set Car,
    safeAreas : set SafeArea
} {
    all u : User | u in users
    all c : Car | c in cars
    all sa : SafeArea | sa in safeAreas
}

sig Ride {
    user : one User,
    car : one Car,
    numberOfPassenger : one Int,
    currentCharge : one Float,
    moneySaveOption : one Bool,
    discountDest : lone Position,
    completed : one Bool,
    dp : one DiscountAndPunish
} {
    car.state = WORKING or car.state = CHARGING or car.state = UNCHARGING
    car.state = WORKING implies completed = False
    car.state != WORKING implies completed = True
    moneySaveOption = True <=> #discountDest = 1
    discountDest in Station.points
    dp.batteryDiscount = True implies completed = True
    dp.rechargeDiscount = True implies completed = True
    dp.distancePunishment = True implies completed = True
    dp.lowBatteryPunishment = True implies completed = True
}

sig DiscountAndPunish {
    passengerDiscount : one Bool,
    batteryDiscount : one Bool,
    rechargeDiscount : one Bool,
    distancePunishment : one Bool,
    lowBatteryPunishment : one Bool
}
```

Part 4

```
fact NoSameCredential {
    no disjoint u1,u2 : User | u1.credential = u2.credential
}

fact NoSamePaymentInfo {
    no disjoint u1,u2 : User | u1.paymentInfo & u2.paymentInfo != none
}

fact NoSameArea {
    no disjoint a1,a2 : Area | a1.points & a2.points != none
}

fact NoCarAtSamePlace {
    no disjoint c1,c2 : Car | c1.currentPosition = c2.currentPosition
}

fact NoSameReservationUserAndCar {
    no disjoint r1,r2 : Reservation | r1.user = r2.user
    no disjoint r1,r2 : Reservation | r1.car = r2.car
}

fact DiscountAndPunishmentCondition {
    no r : Ride | r.dp.passengerDiscount = True and r.numberOfPassenger <= 2
    no r : Ride | r.dp.batteryDiscount = True and r.car.battery != MEDIUM
    no r : Ride | r.dp.rechargeDiscount = True and r.car.currentPosition not in Station.points
    no r : Ride | r.dp.rechargeDiscount = True and r.dp.distancePunishment = True
    no r : Ride | r.dp.distancePunishment = True and r.car.currentPosition in Station.points
    no r : Ride | r.dp.lowBatteryPunishment = True and r.car.battery != LOW
}

//There should be no intersection between {user,car} in Reservation and {user,car} in Ride
fact ReservationRideDistinction {
    all r1 : Reservation, r2 : Ride | r1.user != r2.user and r1.car != r2.car
}

//No user and car can reserve and drive simuteneously
fact NoBusyUserCar {
    all r1 : Reservation, r2 : Ride | r1.user != r2.user
    all r1 : Reservation, r2 : Ride | r1.car != r2.car
    no disjoint r1,r2 : Reservation | r1.user = r2.user
    no disjoint r1,r2 : Reservation | r1.car = r2.car
    no disjoint r1,r2 : Ride | r1.user = r2.user
    no disjoint r1,r2 : Ride | r1.car = r2.car

}
```

Part 6

```
assert AvailableCarPosition {
    no c : Car | c.state = AVAILABLE and c.currentPosition in UnSafeArea.points
}

assert ReservedCarPosition {
    no c : Car | c.state = RESERVED and c.currentPosition in UnSafeArea.points
}

assert LowBatteryPunishmentCheck {
    no r : Ride | r.dp.lowBatteryPunishment = True and r.car.battery = MEDIUM
}
```

Part 7

```
pred example {

/*
    some c : Car | c.state = AVAILABLE
    some c : Car | c.state = RESERVED
   some c : Car | c.state = CHARGING
    some c : Car | c.state = WORKING
    some c : Car | c.state = UNCHARGING
*/

    #Ride = 2

}

run example  for 4


check LowBatteryPunishmentCheck
check AvailableCarPosition
check ReservedCarPosition
```